FBSNG Installation and Administration Guide

Version 1.5

Krzysztof Genser, Tanya Levshina, Igor Mandrichenko

Integrated Systems Department
Fermi National Accelerator Laboratory

Table of Contents

Installing FBSNG 3
- Requirements 3
- FUE Installation.......... 3
- Non-FUE Installation 4

FBSNG Configuration 6
- Daemons Configuration File fbs.cfg 7
  - Set bmgr.......... 7
  - FBSNG Security.......... 9
  - Set jobdb 9
  - Set logd 9
  - Set history..........10
  - Set global..........10
  - Set launcher 11
  - Web Interface 11
- Farm Configuration File farm.cfg..........11
  - Resources Configuration 11
  - Node Classes Configuration..........12
  - Farm Nodes Configuration..........12
  - Process Types Configuration..........13
  - Scheduler Configuration 15

Kerberos Support 16
- Kerberos-based User Authentication..........16
- Creation of Kerberos Credentials for Batch Processes..........17

FBSNG Configuration Utility 18
- Creating Objects 18
- Copying Objects..........19
- Removing Objects 19
- Modifying Object Parameters..........19
- Node Manipulation..........21
- Queue Locking and Unlocking..........21
- Show Commands 21
- Online help..........21

Maintenance Tasks 22
- Starting FBSNG..........22
- System Boot and Shutdown Procedures..........23
- How to Tell Whether FBSNG is Running 24
- Restarting FBSNG Components..........24
- Shutting Down FBSNG Components 25
- Holding and Releasing..........25
- FBSNG Recovery Limitations 25

FBSNG Log Files 27
- Log Clean-up Procedures..........27

Appendix A: Format of FBSNG Configuration Files.......... 28

Appendix B: Sample fbs.cfg file 30

Appendix C: Sample farm.cfg file 32

Installing FBSNG

FBSNG is distributed as a Fermilab Unix Environment (FUE) product. FUE simplifies the installation of FBSNG. However, FBSNG is fully capable to operate in non-FUE environemt. This document covers installation procedures for both FUE and non-FUE **initial** FBSNG installation. Refer to FBSNG Release Notes document if you are **upgrading** FBSNG to a newer version.

Requirements

FBSNG requires the following products to be installed:

- Python 1.5 through 2.1;
- FCSLIB package available from Fermitools must be installed. FCSLIB installation procedure is described below;
- FBSNG GUI requires Python to be built with Tcl/Tk support module Tkinter, and Tcl/Tk must be installed. However, FBSNG GUI functionality is available from command line interface;
- Node resource utilization monitoring tools sysmon and xsysmon included in FBSNG distribution package require rstatd to be installed and running on all nodes to be monitored. Note that FBSNG itself *does not* depend on sysmon/xsysmon or rstatd;
- Kerberos v5 shared libraries and kinit executable are optional. They are required only if it is desired that FBSNG uses Kerberos for user authentication or FBSNG is to create Kerberos credentials for batch jobs.

FUE Installation

Te following is set of steps the administrator has to perform in order to install FBSNG and products it depends on a FUE farm.

1. Using UPD/UPS install and declare FCSLIB product. It is FUE-compliant product distributed through KITS. It is portable across different supported UNIX versions, so it should be declared for NULL UPS flavor. FCSLIB comes with fcslib.table UPS action table file. It must be specified in "ups declare" command. For example:

```
ups declare -0 -c -m fcslib.table -r fcslib/v2_0/NULL \
             fcslib v2_0
```

2. Using UPS/UPD install and declare fbsng product. Currently, fbsng is portable across different known versions of Linux, IRIX and other flavors of UNIX, so only OS flavor, not version has to be included in UPS declaration. FBSNG comes with UPS action table file fbsng.table, which must be mentioned in "ups declare" command. Examples of possible "ups declare" commands are:

```
ups declare -1 -m fbsng.table -r fbsng/v1_5/SunOS -c fbsng v1_5

ups declare -f Linux -m fbsng.table -c \
       -r fbsng/v1_5/Linux fbsng v1_5

ups declare -f IRIX -m fbsng.table -r fbsng/v1_5/IRIX \
       -c fbsng v1_5
```

If you use UPD to install and declare the product, use

```
  upd install -R …
```

3. Create a directory, preferrably in NFS-shared area, for FBSNG configuration files, databases and utility scripts. This directory should be shared by nodes running different flavors of UNIX. This directory will be referred to as <FBSNG_ROOT> or $FBSNG_ROOT.

4. For each UNIX flavor, using UPS, "tailor" the product:

```
  ups tailor -f <flavor> -O <FBSNG_ROOT> fbsng <version>
```

This step must be performed from the same account as used on step 1.

5. Copy directory tree from <FBSNG_DIR>/templates to <FBSNG_ROOT> using:

```
  cp -r <FBSNG_DIR>/templates/* <FBSNG_ROOT>
```

This should create directories:

```
  <FBSNG_ROOT>/cfg        - for configuration
              /bin        - for utility scripts
              /history    - for historical DB
              /logs       - for internal FBSNG logs
              /slog       - for section log files
              /archive    - for compressed log archives
              /jdb        - job database
```

6. Configure FBSNG (see FBSNG Configuration section below). Create FBSNG configuration files fbs.cfg and farm.cfg in <FBSNG_ROOT>/cfg. You can use fbs.cfg.example file in the same directory as an example. Note that interactive FBSNG configuration tools can not be used until BMGR process is running. Therefore, it is recommended to either

- Create initial configuration editing farm.cfg file with a text editor, or
- Rename farm.cfg.template file into farm.cfg, start BMGR (see Starting FBSNG section below), and then use FBSNG Configuration Utility or FBSNG GUI to create desired configuration.

7. Follow instructions in scripts in <FBSNG_ROOT>/bin to adjust them to your installation, in particular, location of setups.(sh|csh) scripts.

Non-FUE Installation

The following procedure should be used to initially install FBSNG on a non-FUE farm.

1. Install FCSLIB product:

   Download fcslib_<version>_<flavor>.tar file.

   Unwind the tar file into, preferably, NFS-shared directory. This directory will be referred to as <FCSLIB_DIR>. FCSLIB contains some Python modules used by FBSNG. It is portable across different flavors of UNIX, and therefore <FCSLIB_DIR> can be shared by all nodes of the farm.

2. Designate a directory for FBSNG configuration files, databases and utility scripts. This directory should be shared by nodes running different flavors of UNIX, therefore, it is recommended to have it in NFS-shared space. This directory will be referred to as <FBSNG_ROOT> or $FBSNG_ROOT.

3. Designate a directory for FBSNG scripts, executables and libraries. This directory will be referred to as <FBSNG_DIR> or $FBSNG_DIR. Each different flavor of UNIX (not each separate version of the same OS) must have its own <FBSNG_DIR>.

4. Download and unwind fbsng_<version>_<flavor>.tar file(s) into corresponding <FBSNG_DIR>s. This will create:

<FBSNG_DIR>/bin	- with FBSNG binaries and Python sources
<FBSNG_DIR>/lib	- with FBSNG API and miscellaneous modules
<FBSNG_DIR>/templates	- with configuration and script templates

5. Copy directory tree from <FBSNG_DIR>/templates to <FBSNG_ROOT> using:

```
cp -r <FBSNG_DIR>/templates/* <FBSNG_ROOT>
```

This should create directories:

```
<FBSNG_ROOT>/cfg        -   for configuration
            /bin        -   for utility scripts
            /history    -   for historical DB
            /log        -   for internal FBSNG logs
            /slog       -   for section log files
            /archive    -   for compressed log archives
            /jdb        -   for job database
```

6. Configure FBSNG (see FBSNG Configuration section below). Create FBSNG configuration files fbs.cfg and farm.cfg in <FBSNG_ROOT>/cfg. You can use fbs.cfg.example file in the same directory as an example. Note that interactive FBSNG configuration tools can not be used until BMGR process is running. Therefore, it is recommended to either
   - Create initial configuration editing farm.cfg file with a text editor, or
   - Rename farm.cfg.template file into farm.cfg, start BMGR (see Starting FBSNG section below), and then use FBSNG Configuration Utility or FBSNG GUI to create desired configuration.

7. Follow instructions in scripts in <FBSNG_ROOT>/bin to adjust them to your installation. In particular, comment out lines related to FUE environment, comment in lines for non-FUE environment, and fill-in actual <FBSNG_DIR> and <FCSLIB_DIR> locations.

fbs_env.(sh|csh) will be 'sourced' by users in order to use FBSNG. Sourcing fbs_env script will add front-end 'fbs' command and FBSNG API to user's PATH and PYTHONPATH respectively. Make sure that <FCSLIB_DIR>/lib is included into PYTHONPATH.

FBSNG Configuration

FBSNG has two configuration files, farm.cfg and fbs.cfg, located in $FBSNG_ROOT/cfg directory.

File fbs.cfg contains configuration parameters for FBSNG components such as TCP/IP addresses, directory locations, time-outs, security-related parameters, etc. This file should be edited by a text editor. For any modifications of this file to take effect, corresponding FBSNG component or components must be re-started.

File farm.cfg contains description of the farm in terms of resources, process types and scheduling parameters. FBSNG comes with FBSNG Configuration Utility. FBSNG monitor includes GUI version of the configuration utility. The configuration utility allows the administrator to perform most of configuration operations without editing farm.cfg file. However, in an emergency situations, and when the configuration utility does not provide necessary functionality, manual editing of farm.cfg file may be required. In this case, BMGR daemon must be restarted after the editing in order for the modifications to take effect.

Manual editing of the farm.cfg is no longer required. All farm configuration functions are available via FBSNG Configuration Utility or FBSNG GUI. Certain rules that must be followed when modifying farm configuration:

- A queue should not be removed until it is empty, i.e. does not have any sections in any state in it. In order to "drain" a queue out, one can lock the queue and wait for all sections to start and complete, or kill all sections in the queue. One can use "fbs queues" or "fbs ls –q <queue>" commands to see whether the queue is empty or not.

  If a non-empty queue is removed from the configuration, BMGR process will not re-start until the queue is returned back into the configuration.

- A process type can be removed only if there are no sections of this type in any state. If necessary, those sections should be killed, or the administrator has to wait for all of them to exit and be removed from FBSNG memory. One can use "fbs ls" command to see what sections currently exist.

  If a process type is removed from FBSNG configuration, BMGR process will not re-start until the process type is returned back into the configuration.

- A node can be removed from the configuration only if no batch processes are running there. A node can be put on hold using "fbs hold node" command in order to block FBSNG from starting new processes on the node. One can use "fbs nodes –l <node>" command to see what processes are running on the node and the node status.

  If a node with batch processes running is removed, the processes will continue to run, but FBSNG will consider them terminated as if the node was shut down.

- A node class can be removed only if no batch processes are running on any of the nodes of the class.

- A resource or resource pool can be removed from the configuration only if there are no running processes or sections in any state that allocate or require the resource or pool. Use “fbs resources” command to see what resources are currently allocated. Unfortunately, there is no easy way to find out what pending sections will require what resources. One can issue “fbs status” command for each pending job or section to find out what resources it requires.
  If a resource or pool is removed, all pending sections that require the resource or pool will remain pending indefinitely.

Both configuration files are described below.

Daemons Configuration File fbs.cfg

FBSNG daemons configuration is specified in $FBSNG_ROOT/cfg/fbs.cfg file. The file has the structure described above in Format of FBSNG Configuration Files section. This file must be located in $FBSNG_ROOT/cfg directory. The following is description of each set of the fbs.cfg file and set parameters.

See Appendix B for an example of fbs.cfg file.

Set bmgr

This set defines parameters for BMGR daemon. The following table describes meaning of each parameter and default values, if any:

Parameter	Value Type	Default	Meaning of Value
api_port	Integer	(required)	TCP server port number for API client. Unless BMGR is running as root, this number should 1024 or greater.
launcher_if_port	Integer	(required)	LauncherIF port where launchers connect to. Unless BMGR is running as root, this number should 1024 or greater.
event_mgr_port	Integer	(required)	TCP server port number for EventManager. Unless BMGR is running as root, this number should 1024 or greater.
host	String	(required)	Host name where BMGR is running
allow_submit	List of Strings	(allow submit from any node)	List of patterns, determines what computers job submission is allowed from. Each pattern can be in one of 3 forms: 1. official node name or IP address(e.g. fnpc21.fnal.gov, 123.2.3.16) 2. Pattern with star (*) at the end or beginning (e.g. 123.2.3.*, *.fnal.gov) 3. Regular expression with either [] or () (e.g. fnpc[a-d].fnal.gov, (fnsfo|fnsfh).fnal.gov)
deny_submit	List of Strings	(empty list)	List of patterns, determines what computers job submission is denied from. Patterns are in the same format as for "allow_submit" parameter.
admin_list	List of Strings	(any user is an admin)	List of user names or numeric user ids. This is list of users authorized to modify FBSNG configuration.
section_log_dir	String	(required)	Directory where section logs are stored
recovery_timeout	Integer	10	BMGR recovery time-out in seconds. During initial recovery interval, BMGR waits for launchers on farm nodes to connect and ignores requests from API clients such as User Interface commands and GUI.
log_ignore	String	""	Log messages not to suppressed 'I','D','E','F' or any combination of them. Each letter suppresses log messages of types Informational, Debug, Error and Fatal respectively.
job_retention_interval	Integer	600	Job retention interval in seconds. Jobs will be kept in memory for the specified time after completion.
short_job	Integer	600	Minimal job time in seconds. Jobs shorter than this will be considered as too short for Scheduler to take into account in fair-share calculation algorithms.
max_sched_run	Integer	None	Maximum time spent by Scheduler in a single uninterrupted run in seconds. If unspecified, Scheduler will continue running until no more sections can be started.

Parameter	Value Type	Default	Meaning of Value
max_node_failures	Integer	None	If number of batch process failures on the same node during last "failure_count_interval" exceeds "max_node_failures", then the node will be automatically held. If either of these two parameters is unspecified, then this feature is turned off.
failure_count_interval	Integer	None	

FBSNG Security

Parameters "allow_submit" and "deny_submit" define what computers job submission is allowed from. These computers are referred to as "trusted computers". Submission is allowed from the node if:

1. Its IP address or official IP node name matches any pattern from allow_submit list, or the list is empty

and

2. Neither its IP address nor its official IP node name matches any pattern on the deny_submit list, or the list is empty.

Together with allow_submit and deny_submit lists, the admin_list parameter defines list of "trusted users" or "administrators". The user is an administrator if:

1. It is on one of "trusted" computers

and

2. Its UID or username is on the admin_list, or the admin_list is empty.

Depending on username, UID and the node the user comes from, he/she can perform the following actions:

Trusted node ?	Trusted user ?	Can modify configuration	Can submit jobs
Yes	Yes	Yes	Yes
Yes	No	No	Yes
No	Yes	No	No
No	No	No	No

Any user on any computer is allowed to obtain run-time and configuration information.

Set jobdb

This set controls where FBSNG keeps its job data database. Currently, it has the only parameter:

Parameter	Value Type	Default	Meaning of the value
root	String	(required)	Path to the directory where job database is located. The directory should not contain any other information than the database created by BMGR

Set logd

This set defined parameters for FBSNG logd daemon and supplemental archiving, compression and purging scripts.

Parameter	Value Type	Default	Meaning of the value
host	String	(required)	Log daemon host name
server_port	Int	(required)	Log daemon UDP port number
log_dir	String	(required)	Log directory
email_wait	Int	1800	Time interval between sending identical e-mail messages in seconds (default=30 minutes)
email_command	String	"Mail"	Command to be used to send e-mail
arch_dir	String	(required)	Log archive storage directory
days_to_zip	Int	1	Age of log files to compress in days
days_to_archive	Int	5	Age of files to archive in days
ignor_codes	String	""	Message codes to ignore. This is a combination of single-character message type codes. Any message with code present in this string will be discarded without storing in the log files.
admin_address	String	None	e-mail address for error reports

Set history

This set defines parameters for FBSNG job history database:

Parameter	Value Type	Default	Meaning of Value
archive_period	Integer	1	Interval between moving the data from history file to archive file, in days.
hist_dir	String	(required)	Directory where history file(s) are stored
archive_file	String	archive.log	Archived history file name
hist_file	String	hist.log	Current history file name
compress_period	Integer		Interval in days between compressing history information into .tar.Z files
days_to_store	Integer		Time in days to keep information in history database

Set global

This set has two parameters:

Parameter	Value Type	Default	Meaning of Value
domain	String	(empty)	Common suffix for farm node names
mail_command	String	/bin/mail	External command to be used by FBSNG components to send e-mail messages
farm_name	String	(empty)	Name of the farm

Value of “domain” parameter should be common IP domain name for all farm nodes. If specified, farm node name used by FBSNG will be determined by truncating the specified domain name from the “official” (obtained by gethostname() system function) IP node name.

Set launcher

Set of type “launcher” has two parameters, “stat_port” and optional “k5_kinit_command”. Value of “stat_port” parameter is UDP port number used by API clients to obtain run-time information about running processes. “k5_kinit_command” parameter is described in “Kerberos Support” section.

Web Interface

Version 1.4 includes its own HTTP server which provides basic monitoring functionality. HTTP daemon is a separate process called “fbswww”. It is configured in set “fbswww” in fbs.cfg file. It has 2 optional parameters:

Parameter	**Value Type**	**Default**	**Meaning of Value**
port	Integer	8080	FBSWWW HTTP server port number
graphs_dir	String	(none)	Top directory of historical performance graphs storage

If graphs directory is specified, it is expected that it has 2 subdirectories:

```
<root>/display
<root>/print
```

First will contain graphs to be displayed by browser and second – graphs to be formatted for printing. Graph file names must be in the form *_<period>.(gif|jpg). Where period is “day”, “week”, “2weeks”, “month”, “3months” or “year”.

URL for the web interface is

```
http://<node>:<port>/fbswww
```

Farm Configuration File farm.cfg

This file contains definition of farm resources, farm nodes, their classes, process types, and queues. See FBSNG Resources and FBSNG Scheduler documents for more details on these concepts and relationships between them. This file must be located in $FBSNG_ROOT/cfg directory. The file protection mask must allow BMGR process to write into this file. See Appendix C for an example of farm.cfg file.

Normally, FBSNG Configuration Utility or GUI should be used to modify farm configuration defined in farm.cfg file. However, under certain circumstances it may be necessary to edit this file using a text editor. In such cases, BMGR daemon must be re-started in order for the modifications to take effect.

Resources Configuration

Farm resources are defined in set “resources” of farm.cfg. The set has two parameters:

Parameter	**Value Type**	**Default**	**Meaning of Value**
local	List of Strings	(no local resources)	List of local resource names, including node attributes.

Parameter	Value Type	Default	Meaning of Value
global	Dictionary String:Integer	(no global resources)	Dictionary with resource names as keys and corresponding integer resource capacities as values

Resource pools are defined in set "resource_pools". Each pool is represented with single set parameter with the pool name as parameter name and list of underlying resource names as value.

Node Classes Configuration

Each node class is represented as a separate set of type "node_class" and the class name as set id. Each set has two parameters:

Parameter	Value Type	Default	Meaning of Value
resources	Dictionary String:Integer	(no local resources defined for the class)	Dictionary with local resource names as keys and optional integer capacities as values. If a value is not specified, corresponding resource will be considered as a node attribute.
local_disks	Dictionary String:String	(no local disks defined for the class)	This dictionary describes correspondence between resource names used to represent local scratch disks and actual locations of the scratch disk storage. For each key-value pair, the key must be the name of one of the local resources defined for the node class, and the value – path to corresponding directory.
Power	Floating point number	1.0	Relative computer power.

Local scratch disk areas are specified as their top directories. If the specified top directory does not exist, it will be created. FBSNG assumes that the specified directory will not contain any file or directory other than those created by FBSNG as temporary working directories for batch processes, and therefore, can delete any file or directory found there at any time.

If the same batch process takes T(X) and T(U) time to complete on nodes of class X and nodes of a class with power=1.0 respectively, then relative computer power of nodes of class X is defined as:

Power(X) = Time(U)/Time(X)

FBSNG Scheduler uses the computer power to account more accurately for resource utilization and when time limits are applied. As a first approximation, computer power can be set proportional to computer clock speed. If more accurate calculations are required, some benchmarking may be necessary. One should keep in mind that different applications can be running more or less efficiently on a given computer with given OS parameters, and therefore computer power, calculated using different applications can differ slightly.

Farm Nodes Configuration

Set "node_list" defines the node class each node belongs to. The set consists of

```
     <node name> = <node class name> [hold <hold reason>]
```

lines, one line per node. Node name for a farm node is defined as its official IP node name with domain name (as specified in set "global" of fbs.cfg, if any) truncated.

If the node is put on hold, node class name will be followed by keyword "hold" and arbitrary (multiword) string specifying the hold reason.

Process Types Configuration

Each process type is represented as a separate set of type "proc_type" and the process type name as the set id. Each set has 3 patameters:

Parameter	Value Type	Default	Meaning of Value
proc_rsrc_defaults	Dictionary String:Integer	No default resource requirements	This dictionary defines default resource requirements for processes of this type. For each key-value pair, the key must be the name of one of a resource, and the value – integer amount of the resource required. For node attributes, the value should be omitted.
sect_rsrc_defaults	Dictionary String:Integer	No default resource requirements	This dictionary defines default resource requirements for sections of this type. For each key-value pair, the key must be the name of one of global resources, and the value – integer amount of the resource required.
resource_quota	Dictionary String:Integer	No local disks defined for the class	This dictionary resource allocation quotas for the process type. For each key-value pair, the key must be the name of one of local or global resources, and the value – the allocation limit. If a resource is not mentioned in the dictionary, the process type can allocate any amount of the resource.
users	List of strings	Anybody	List of users who are allowed to use this process type. "*" means anybody, "-" means nobody.
nodes_allow	List of strings	All nodes	List of nodes processes of this type are allowed to run on. "*" means any node, "-" means empty list.
nodes_disallow	List of strings	(empty)	List of nodes processes of this type are not allowed to run on. "*" means any node, "-" means empty list.
max_prio_inc	Integer	No limit	Maximum allowed increment of a section priority
max_nodes	Integer	No limit	Maximum number of nodes processes of this process time are allowed to run on at any given time
realtime	Integer	No limit	Maximum allowed process run time, seconds
cputime	Integer	No limit	Maximum allowed CPU time consumed by individual batch process
nice	Integer	0	Base NICE level for processes of the process type

FBSNG process type can be protected from unauthorized use. This is done by defining list of users allowed to use the process type either by editing farm.cfg file, or using FBSNG Configuration Utility or GUI. Authorized users have some control over the process type. In particular, they are allowed to:

- Modify lists of allowed and disallowed nodes
- Modify maximum priority increment

When FBSNG chooses an available node to start a batch process of certain process type, it uses lists of allowed and disallowed nodes in the following way:

- If the disallowed node list contains "*", the node will be considered only if its name is in allowed list

- Otherwise, if the allowed list contains "*", the node will not be considered if it is listed in the disallowed list
- Otherwise, the node will be considered only if it is listed in the allowed list but not in the disallowed list

If both queue where the job section is submitted to and the process type of the section have real or CPU time limit defined, the lesser one will be used.

If "nice" parameter is defined for the process type, then that NICE level will be used for all processes that start under this process type. If user defines NICE in the job description, the value specified by the user will be *added* to the value defined for the process type. Negative values are not accepted in job description and will be ignored.

Scheduler Configuration

FBSNG Scheduler is configured by defining set of Scheduler queues and their parameters. See FBSNG Scheduler document for more details on its algorithm and configuration parameters.

In farm.cfg file, each queue is represented by one set of type "queue" with the following parameters:

Parameter	**Value Type**	**Default**	**Meaning of Value**
proc_type	String	(none)	Default process type for the queue
s_prio_max	Int	100	Maximal section priority
s_prio_gap	Int	1000	Section priority gap
q_prio_min	Int	0	Minimal queue priority (= initial queue priority)
q_prio_max	Int	100	Maximal queue priority
q_prio_dec	Int	10	Queue priority decrement
q_prio_gap	Int	1000	Queue priority gap
cputime	Int	(unlimited)	Process CPU time limit
realtime	Int	(unlimited)	Process run time limit
status	Strings	OK	Either "locked" or "held" or both
users	List of Strings	(any user)	List of users allowed to use the queue. "*" means anybody, "-" means nobody.
Ptypes	List of Strings	(any process type)	List of process types allowed in the queue. "*" means any process type, "-" means empty list.

FBSNG queue can be protected from unauthorized queue by defining list of users who are allowed to submit job sections to the queue. Also, the administrator can list process types of the sections allowed to be in the queue. This can be done by either editing farm.cfg file, using FBSNG Configuration Utility or FBSNG GUI.

Authorized users have some administrative control over the queue. They are allowed to:

- Hold/release the queue
- Modify maximum/minimum section priority
- Modify section priority gap

Kerberos Support

FBSNG has Kerberos support capabilities. On farms where Kerberos is used to implement strong authentication methods, FBSNG can be configured to interact with Kerberos to acheve one or both of the following:

- Kerberos-based FBSNG user authentication
- Creation of initial Kerberos credentials for batch processes

In order to use any of these features, "kerberized" version of FBSNG must be installed and these features must be turned on in the FBSNG configuration.

Kerberos-based User Authentication

In order to use Kerberos for FBSNG user authentication, the administrator must add "kerberos" set to fbs.cfg file. The "kerberos" set consists of the following parameters:

Parameter	**Value Type**	**Default**	**Meaning of Value**
client_auth_required	String "yes" or "no"	"no"	Whether Kerberos should be used for user authentication.
principal	String	(required)	Name of FBSNG Kerberos service principal
keytab	String	(required)	Path to FBSNG' principal keytab file

Kerberos will be used for user authentication if the "client_auth_required" field in "kerberos" set has value "yes". In this case, the set must contain name of the FBSNG service principal and path to keytab file with service password. The keytab must be readable by BMGR process.

Kerberos-based user authentication, if turned on, will be used only when a user submits a batch job or is trying to modify FBSNG configuration. All other operations such as obtaining run-time information do not require user authentication, and therefore can be performed even by a user without any Kerberos credentials.

FBSNG supports the concept of group accounts. Kerberos services such as telnet, rsh, etc. allow a user with valid Kerberos credentials to log in to another (group) account without gaining credentials for this account. For example, one can set up an account named "group123" and create a .k5login file in its home directory with list of Kerberos principals allowed to use this account. If such file allows user "alice" to log in under "group123" account, it is possible for user "alice" to have UNIX identity of "group123" and have only "alice" Kerberos credentials. FBSNG can be configured so that UNIX user "group123" with Alice's Kerberos credentials will be able to submit and run batch jobs under "group123" account. This is configured with parameter "principals" of set "user_profile" in fbs.cfg file. Value of this parameter is a space-separated list of templates for Kerberos principals allowed to "impersonate" the user. The template is compared to actual Kerberos principal of the user who is attempting to submit a job by applying the following rules:

- If the first element of the list is "+", users credentials will be accepted if they match either any of the remaining elements, or any of elements of the default list found in unnamed "user_profile" set
- If the first element of the list is "-", user credentials will be accepted if they match an element of the default list, but do not match any of the remaining elements of the list
- "%u" is substituted with the username
- Star ("*") is interpreted as a wildcard symbol

For example:

```
%set user_profile         # default set, common for all users
# require that the user is authentiacated by Kerberos
principals = %u@the.realm

%set user_profile group123     # this is group account
# allow Alice and Bob to use it
principals = + alice@the.realm alice/*@the.realm
       bob@the.realm bob/*@the.realm

%set user_profile test    # this is test account
# require only that this person has some Kerberos credentials
# from our realm
principals = *@the.realm
```

Another example:

```
%set user_profile         # default set, common for all users
# generally, the only requirements is to have any credentials
# from our realm
principals = *@the.realm

%set user_profile farmprod
# this is production account, accept only Alice from any
# realm
credentials = alice@*

%set user_profile farmtest
# this is test account, accept anybody but Bob
credentials = - bob@* bob/*@*
```

Modifications of values of the parameters described here will take effect only after BMGR process is restarted.

Creation of Kerberos Credentials for Batch Processes

FBSNG, if so configured, can create Kerberos credentials for batch processes so that batch processes can use "kerberized" services. In order to use this feature, system administrator must

- create Kerberos principals for those users who wish use "kerberized" services in the context of batch processes
- create keytab files for these principals
- make keytab files available on the farm nodes and readable by "root" user

In addition, the following parameters must be specified in fbs.cfg file.

Parameter "k5_kinit_command" in set "launcher" specifies what command must be issued to create credentials for a batch processes. The command is specified in the form of a template. All occurrences of string "%u" will be replaced by FBSNG with actual UNIX batch process user name. For example, "k5_kinit_command" template might look like this:

```
kinit -k -r 96h -t /var/adm/%u.keytab %u/thefarm@therealm
```

When this template is used for user "farms", actual command will look like this:

```
kinit -k -r 96h -t /var/adm/farms.keytab farms/thefarm@therealm
```

Set "user_profile" can be used to specify whether or not credentials must be created for processes of each user. The administrator can create one default set without set id, and then create user-specific set for each user using username as the set id. The "user_profile" set has parameter named "create_k5_credentials". This parameter can have one of 3 values:

- "required" – credentials must be created for processes of this user. If for some reason the credentials could not be created, the node must be automatically put on hold, and user process should be considered failed.
- "optional" – FBSNG will attempt to create credentials. Failure will be reported to the user, the batch process will be allowed to proceed.
- "no" – FBSNG will not attempt to create credentials for this user.

It is necessary to restart Launcher processes on farm nodes for "create_k5_credentials" and "k5_init_command" parameter value modifications to take effect.

FBSNG Configuration Utility

The utility can work in 2 modes, interactive and single command. In interactive mode the utility prompts for next command after completion of previous. In single-command mode, it executes only one command and exits. Interactive mode is invoked by "fbs config" command. Command syntax is:

```
<command> <arguments> ...
```

Single-command mode is involed by typing

```
fbs config <command> <arguments> ...
```

Both modes accept the same set of commands and arguments:

Creating Objects

The following commands are used to create FBSNG objects:

Creating a Queue

```
create queue <queue name> [<def. process type>] [<parameters>]
```

See "set queue" command description for list of optional parameters. If no parameters are specified, the queue is created with default set of parameters and the specified process type associated with it. Queues are always created in locked state. In order to unlock a queue, use "unlock queue" command.

Creating a Node Class

```
create  nclass  <node class name>
```

The command creates an empty (having no nodes associated to it) node class with no resources or local disks defined. Use "set nclass" command to configure local resources and local scratch disks for the node class. Use "add/remove node" commands to manipulate list of nodes of the class.

Creating a Process Type

```
     create  ptype   <proc type name> [<parameters>]
```

See "set ptype" command description for accepted list of optional parameters. If no parameters are specified, the new process type will have no quotas or default resource requirements defined.

Note that when a process type is created, its authorized users list is set to be empty. Administrator has to use "set ptype … users …" command to explicitly allow some or all users to use new process type.

Creating a Resource Pool

```
     create  pool    <pool name> <resource name> ...
```

The command creates new resource pool combining listed resources.

Creating a Global Resource

```
     create  gr      <global resource name> <capacity>
```

The command creates new global resource with specified capacity.

Creating a Local Resource

```
     create  lr      <local resource name>
```

The command creates new local resource.

Copying Objects

FBSNG Configuration Utility can be used to create new objects using existing objects as templates so that new objects have the same parameters as existing ones. Common syntax for this command is:

```
     copy (queue|ptype|nclass) <existing object name> <new name>
```

The command will make a copy of a queue, process type or node class depending on first argument of the command.

Removing Objects

To remove one or node, queue, process type, node class, resource or resource pool, one can use "remove" command:

```
     remove (node|pool|gr|lr|ptype|nclass|queue) <name> …
```

An object can be removed only if it is not in use. For example, a queue can be removed only if it is empty, process type can be removed only if there are no sections in the system of this process type, and so on. FBSNG performs all necessary checks to preserve referential integrity of FBSNG configuration.

Modifying Object Parameters

The following commands can be used to modify parameters of existing objects

Setting Queue Parameters

```
     set queue <queue name> <parameter>[:<value>] ...
```

The command accepts the following parameters: QPGap, QPInc, QPDec, MaxQPrio, MinQPrio, MaxSPrio, SPGap, Prio, DefProcType, RealTimeLimit, CPUTimeLimit,

Values for parameters RealTimeLimit and CPUTimeLimit can be omitted together with separating colons. In this case corresponding limits will be removed.

To modify list of authorized users or allowed process types use

```
     set queue <queue name> users (-|*|<user> …)
     set queue <queue name> ptypes (-|*|<proc. type> …)
```

In both cases, dash "-" means empty list and star "*" means "any":

Setting Node Class Parameters

```
     set nclass  <node class> rsrcs <resource>:<capacity> ...
```

"rsrcs" keyword indicates that local resource capacity for the class should be set as specified as by the following dictionary.

```
     set nclass  <node class> disks
                  <disk resource name>:<root directory> ...
```

"disks" keyword indicates that the following is the mapping between resource names and associated local scratch disk locations. Local disk resources must already be defined as local resources for the node class.

Setting Process Type Parameters

```
     set ptype   <proc type name>    <parameter> <value>
```

Parameters and their values are:

- srdef   <resource>:[<amount>] ... – default section resource requirements
- prdef   <resource>:[<amount>] ... – default process resource requirements
- rquota <resource>:<amount> ... – resource allocation quota
- maxpinc <value> - maximum allowed priority increment
- users (-|*|<user> …) – list of authorized users of the process type
- +nodes (-|*|<node> …) – list of nodes processes of the type are allowed to run on
- -nodes (-|*|<node> …) – list of nodes processes of the type are not allowed to run on
- maxnodes (-|<value>) – maximum number of nodes processes of this type are allowed to run on. "-" means no restrictions.
- realtime (-|<number>[d|h|m|s]) – elapsed time limit for processes of this type. "-" means no limit. The number may be followed by "d","h","m" or "s" which stand for "days","hours","minutes" and "seconds" respectively. Default is "seconds".
- cputime (-|<number>[d|h|m|s]) – CPU time limit for processes of this type. "-" means no limit. The number may be followed by "d","h","m" or "s" which stand for "days","hours","minutes" and "seconds" respectively. Default is "seconds".

In users and node lists specifications, star "*" means "any", and dash "-" means empty list.

Note that in cases when both Queue and Process Type impose a time limit on a batch process, the lowest time limit will be used.

Setting Global Resource Capacity

```
     set gr      <global rsrc name>  <capacity>
```

Setting Composition of a Resource Pool

```
     set pool <pool name> <resource name> ...
```

Node Manipulation

The following commands should be used to add or remove nodes to or from an existing node class. Only nodes without any processes running can be removed from a node class. When a node is added to a node class, its status becomes "held".

```
     add node <node class name> <node name> ...
     remove node <node name> ...
```

Queue Locking and Unlocking

An administrator can lock a queue to disable submission of new sections to it in order to "drain" the queue down. When a new queue is created, it must be unlocked before it can be used. The commands are:

```
     lock queue (all|<queue name>)
     unlock queue (all|<queue name>)
```

If the keyword "all" is used instead of a queue name, the operation will be applied to existing queues.

Show Commands

In order to see information about one or more objects of certain type, one can use "show" command:

```
     show    (queue|nclass|pool|ptype|rsrc) [<name> ...]
```

If no object name is specified, all objects of the type will be displayed. "rsrc" keyword is used to show information about all types of resources.

Online help

If a user uses incomplete command, usage information is printed. Also a user can issue help command for a brief command summary.

Maintenance Tasks

Starting FBSNG

FBSNG has the following permanently running processes (daemons):

- bmgr - runs on preferably most reliable node of the farm. This daemon can run under non-privileged account. It must have write access to log, slog, archive, history, cfg subdirectories under <FBSNG_ROOT> and files in these directories, and have at least read access to the rest of <FBSNG_ROOT> subtree.

- launcher - an instance of this process must run on each worker node of the farm. Launcher must run as root and must have read access to files in <FBSNG_ROOT>/cfg.

- logd - one instance, runs on preferably most reliable node of the farm. This daemon can run under non-privileged account. It must have write access to <FBSNG_ROOT>/log subdirectory, and have at least read access to the rest of <FBSNG_ROOT> subtree. Logd is not required for FBSNG to work properly. It is used only to store FBSNG internal logs. Logd and bmgr do not have to run on the same node.

- fbswww – one instance on any computer inside or outside of the farm. Typically it should run on the same node where bmgr is running, but that is absolutely unnecessary. This daemon can run under non-privileged account.

Although not necessary, it is recommended to set up unprivileged account for bmgr and logd and create <FBSNG_ROOT> directory in NFS-exported home area of such account.

In order to start each daemon, administrator should use start_bmgr.sh, start_logd.sh and start_lch.sh scripts found in <FBSNG_ROOT>/bin directory. Before they can be used, the scripts must be adjusted to the particular installation following instructions found in the script files. Administrator must 'su' to proper account before using the scripts. The scripts are designed to be called from system boot-time start-up scripts.

FBSNG distribution includes "sendcmd" script. This script, located in $FBSNG_ROOT/bin directory, can be used to perform certain operations on a number of nodes using "rsh" command. "sendcmd" script can be used to start launcers on farm nodes in the following way:

```
cd $FBSNG_ROOT/bin
./sendcmd fnpc 101 120 /home/farms/fbsng_root/bin/start_lch.sh
```

This command will run the start_lch.sh script on nodes fnpc101 through fnpc120.

In cases when using start_xxx.sh scripts is not acceptable due to site-specific policies or procedures, FBSNG Administrator can use lower level commands to start FBSNG components. The following commands run corresponding daemons:

```
fbs bmgr            # run BMGR
fbs launcher        # run Laucnher
fbs logd            # run Logd
fbs fbswww          # run FBSWWW HTTP server
```

System Boot and Shutdown Procedures

Usually, FBSNG components start at system boot time. Also, BMGR process needs to be shut down gracefully before system shut down. The following are examples of scripts to be used on FBSNG central and worker nodes during UNIX boot and shut-down:

```
#!/bin/sh
# Script to be placed into /etc/init.d directory on FBSNG central
# node.
#
# Here “farms” is the account to run FBSNG BMGR and Logd under,
# /home/farms/fbsng is FBSNG_ROOT directory
#
# At system boot time, BMGR and Logd start as “farms”, and Launcher
# starts as “root”.
#
# At system shutdown, only BMGR needs to be shut down gracefully

case "$1" in
'start')
                su farms -c "/home/farms/fbsng/bin/start_bmgr.sh"
                su farms -c "/home/farms/fbsng/bin/start_logd.sh"
                /home/farms/fbsng/bin/start_lch.sh
                exit 0
        ;;

'stop')
                su farms -c "/home/farms/fbsng/bin/shutdown_bmgr.sh"
                exit 0
        ;;

*)
        echo "usage: $0 {start|stop}"
        ;;
esac

#!/bin/sh
# Script to be placed into /etc/init.d directory on FBSNG worker
# nodes.
#
# Here /home/farms/fbsng is FBSNG_ROOT directory
#
# At system boot time, Launcher starts as “root”.
#
# Launchers do not need to be shut down gracefully

case "$1" in
'start')
                /home/farms/fbsng/bin/start_lch.sh
                exit 0
        ;;

'stop')
                exit 0
        ;;

*)
        echo "usage: $0 {start|stop}"
        ;;
esac
```

Templates of these scripts are included in FBSNG distribution in templates/bin subdirectory. They are named “start_fbsng_central.sh” and “start_fbsng.sh”.

How to Tell Whether FBSNG is Running

First quick check is to use UNIX “ps” command. Here is what ps output should look like for BMGR:

```
fnpcb> ps axw | grep bmgr
17177 pts/12   S      0:01 sh -f /home/farms/fbsng_v1_1/bin/fbs_run.sh bmgr
20163 pts/12   S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng bmgr
20164 pts/12   S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng bmgr
20165 pts/12   S      0:09 python bmgr.py
```

for launcher:

```
fnpcb> ps axw | grep launcher
13660 ?        S      0:00 sh -f /home/farms/fbsng_v1_1/bin/fbs_run.sh launcher
20488 ?        S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng launcher
20491 ?        S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng launcher
20509 ?        S     28:50 python launcher.py
```

and for logd:

```
fnpcb> ps axw | grep logd
18921 pts/12   S      0:00 sh -f /home/farms/fbsng_v1_1/bin/fbs_run.sh logd
19007 pts/12   S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng logd
19008 pts/12   S      0:00 sh /fnal/ups/prd/fbsng/v1_1/Linux/bin/fbsng logd
19010 pts/12   S      0:21 python logd.py
```

Also, in order to see if FBSNG is working properly, administrator can use “fbs nodes” command. If the command prints list of the farm nodes, it means that important FBSNG components are configured properly, and BMGR process is running. Here is what normal “fbs nodes” output might look like:

```
fnpcb> fbs nodes
 HOST               STATUS   CLASS       PROCESSES
 ----               ------   -----       --------
 fnpcb              up       IO          0:
 fnpc104            down     Worker      0:
 fnpc105            down     Worker      0:
 fnpc106            down     Worker      0:
 fnpc107            down     Worker      0:
 fnpc101            down     Worker      0:
 fnpc102            down     Worker      0:
 fnpc103            down     Worker      0:
 fnpc108            down     Worker      0:
```

It should be noted that after starting, BMGR process is supposed to ignore all request while it performs initial recovery functions. The recovery time interval is defined in fbs.cfg file.

“fbs nodes” command will print status of all farm nodes. “down” node status may indicates that either the node is down, or launcher process on the node can not connect to BMGR process, or the launcher is not running there.

Restarting FBSNG Components

Scripts kill_lch.sh, kill_bmgr.sh and kill_logd.sh found in <FBSNG_ROOT>/bin directory should be used to restart launcher, BMGR and logd processes respectively.

Of course, they must be invoked on the node where the process is running, and require necessary privileges to kill the process. Utility script "sendcmd" distributed with FBSNG can be used to restart FBSNG components remotely, for example:

```
cd $FBSNG_ROOT/bin
./sendcmd fnpc 101 120 /home/farms/fbsng_root/bin/kill_lch.sh
```

This will kill (and then automatically restart) launcher processes on nodes fnpc101, fnpc102, … fnpc120.

Shutting Down FBSNG Components

In order to shut down FBSNG component so that it does not re-start afterwards, shutdown_lch.sh, shutdown_bmgr.sh and shutdown_logd.sh scripts found in <FBSNG_ROOT>/bin directory should be used. Utility script "sendcmd" distributed with FBSNG can be used to restart FBSNG components remotely, for example:

```
cd $FBSNG_ROOT/bin
./sendcmd fnpc 101 120 /home/farms/fbsng_root/bin/shutdown_lch.sh
```

This will shutdown launcher processes on nodes fnpc101, fnpc102, … fnpc120.

Holding and Releasing

FBSNG queues and nodes can be held. If a queue is held, no new jobs will start from it. If a node is held, no new batch processes will start on the node. Holding of queues or nodes has no effect on running sections or processes, therefore, holding of queues or nodes can be used to "drain" the system down, for example, before planned shut down. If an object is held, it can be released which returns it back to normal state. The following commands can be used by an administrator to hold and release objects:

```
fbs hold queue <queue name> …         # hold specific queues
fbs hold queue all                    # hold all queues
fbs hold node <node name> …           # hold specific nodes
```

Corresponding "release" commands are:

```
fbs release queue <queue name> …      # release specific queues
fbs release queue all                 # release all queues
fbs release node <node name> …        # release specific nodes
```

Under certain circumstances, FBSNG itself can hold some nodes. This happens, for example, when FBSNG discovers that user's home area does not exist on the node. In such cases, after the problem is investigated and fixed, before releasing the node, it is necessary to restart Launcher process on the node.

FBSNG Recovery Limitations

FBSNG is designed so that it can recover from most of planned or unexpected component failures. However, when planning re-start of FBSNG components, for example after modifying FBSNG configuration, the administrator should follow certain rules:

Maintenance Tasks

Launcher on a farm node should never be re-started while any batch processes run on the farm node. Launcher termination is interpreted as shutting down the node it runs on, and any processes running there are considered to be terminated. If it is necessary to shut down or re-start a launcher, the administrator should hold the node using "fbs hold node" command, either kill all jobs or job sections running on the node, or wait until all processes running on the node exit, and only then re-start or shut down the launcher process.

In order to prevent FBSNG database corruption, BMGR process **must not** be killed with SIGKILL (usually signal 9). It should be killed only with SIGINT signal. It may take several seconds for BMGR to exit after receiving the signal depending on what state it was when the signal was received. It is recommended to always use kill_bmgr.sh to re-start the BMGR process or shutdown_bmgr.sh to shut it down permanently. As long as it is done properly, it is safe to shut down or re-start BMGR process at any time. When BMGR process re-starts, it will recover run-time information about pending and running batch jobs.

Logd process can be killed virtually any time. If logd re-starts shortly, no log information will be lost. However, if it does not re-start soon enough, some old log information may be discarded.

FBSNG Log Files

Unless configured otherwise, FBSNG keeps its own log files in <FBSNG_ROOT> directory tree. They are organized as follows:

```
<FBSNG_ROOT>/logs/bmgr    -     BMGR log files
<FBSNG_ROOT>/logs/lch     -     Launcher log files
<FBSNG_ROOT>/slog         -     Section log files
```

BMGR and Launcher log file names include date when the file was open and the node where the process is running. For Launcher, the log file name includes UNIX process ID of the Launcher process. The log files are closed around midnight every day and new log file with new date encoded in it is created. Examples of BMGR and Launcher log files are:

lch.fnpc105.941.20000628.log – this log file contains messages received from the Launcher running on fnpcb computer with process ID 941 during the day of 6/28/2000.

bmgr.fnpcb.20000705.log – contains messages received from BMGR running on fnpcb during the day of 7/5/2000.

These log files are created by logd process, and therefore would not exist if the logd process was not running.

Section log files are created by BMGR process itself and can be viewed using "fbs slog <section id>" command

Along with log files maintained by logd, BMGR and Launchers create stderr and stdout files in <FBSNG_ROOT>/bmgr and /fbsng/tmp directories respectively on the nodes where they are running. These files usually contain only Python stack crash dump information and should be consulted if corresponding log file is unavailable or does not contain necessary information.

Log Clean-up Procedures

FBSNG distribution package includes set of scripts to be used to maintain FBSNG logs and purge old log files. The scripts are designed to run periodically, e.g. daily, as cron tasks. The scripts are located in <FBSNG_DIR>/templates/bin and should be copied to <FBSNG_ROOT>/bin directory during installation. The scripts are:

logd.run.cron – this script performs the following functions:
- compresses FBSNG and section log files older than one or two days old
- removes debug messages from old log files, then archives them and compresses the archive files producing .tar.Z files and stores them into archive directory.

The script parameters can be customized in fbs.cfg in set "logd".

history.run.cron – this script is obsolete and should be removed from crontab.

Appendix A: Format of FBSNG Configuration Files

FBSNG configuration consists of two plain text files, fbs.cfg and farm.cfg. Both files have the same general structure. Each file consists of *sets* of parameters. Each set has *set type* and *set identification* (*set id)*. There should be only one set with the same combination of set type and set id in the file. Each set begins with *set header* line:

```
%set <set type> [<set id>]
```

Set id is optional, and if not specified, the set is called *type default set*. There should be not more than one default set for each type. Values of all parameters specified in the type default set will be used as defaults for all other sets of the same type.

For example, the following configuration specification

```
%set car
wheels = 4
seats = 5

%set car van
seats = 8

%set car convertible
convertible_top = yes
```

is equivalent to:

```
%set car van
wheels = 4
seats = 8

%set car convertible
wheels = 4
seats = 5
convertible_top = yes
```

Set header line is followed by one or more parameter definitions. Parameters can be specified in one of 3 formats:

```
<name> = <value>
<name> = <item> <item> …
<name> = <key>[:<value>] <key>[:<value>] …
```

First format is used to represent a parameter with single value, second – a parameter with a list of values and third – a dictionary. In case of dictionary type, some values may be omitted together with separating colons. For the keys without values, the application will use some default value.

In case when a list or a dictionary does not fit into single line, it can be continued on the next line. For example:

```
%set car
tires = left-front:34psi left-rear:30psi
       right-front:34psi right-rear:30psi
       spare:unknown
```

Words in parameter values in first two formats can be enclosed in single or double quotes to allow white space inside a word. Pound sign (#) can be used as a comment character. Except for when pound sign is in the middle of a word, right end of the line starting with the pound sign is ignored. For example:

```
%set parser                     # this set describes some
                                # sort of parser
special_chars = +-#*/           # Pound in the middle does NOT
                                # begin comment
error_format = 'ERROR: %t%d'    # use quotes to enter spaces
#uncomment to allow a time-out
#time_out = 10
```

Appendix B: Sample fbs.cfg file

This sample file is supplied with FBSNG distribution package.

```
#
# fbs.cfg - FBSNG configuration file example
#

%set global
domain = fnal.gov           # common domain name for farm nodes only
#farm_name = "CDF Farm"     # name of the farm
#mail_command = /usr/lib/sendmail
                            # command to be used to send e-mail

#------------------------------------------------------------------
# Define FBS BMGR process parameters:

%set bmgr
api_port = 6667                 # API server TCP port number
event_mgr_port = 6669           # Event Manager TCP server port number
recovery_timeout = 10           # recovery time-out in seconds
launcher_if_port = 5557         # Launcher interface TCP port number
host = hppc.fnal.gov            # IP address of BMGR node
                                # domain name from "global" does not apply
                                # here !
job_retention_interval = 20     # How long completed jobs are to be kept
                                # in memory (seconds)
section_log_dir = /home/farms/fbsng/slog
                                # where section logs are to be stored
#allow_submit = *.fnal.gov      # where to accept job submission from
#    131.225.*
#deny_submit = fnpc[a-c].fnal.gov   # where to reject submits from
#admin_list = root farms        # list of users allowed to re-configure
                                # the farm
log_ignore = D                  # do not send debug messages to logd
short_job = 60                  # in seconds, jobs shorter than this
                                # are considered too short to be taken
                                # into account for fair-share calculations

#------------------------------------------------------------------
# Define FBSWWW server parameters

%set fbswww
port = 8080
graphs_dir = /tmp/fbswww_graphs


#------------------------------------------------------------------
# Define Log Daemon parameters

%set logd
host = hppc.fnal.gov            # IP address of the node where it is running
                                # domain name from "global" does not apply
                                # here !
server_port = 7654              # UDP server port number
log_dir = /home/farms/fbsng/logs
                                # Root directory for log storage
arch_dir = /home/farms/fbsng/archive
                                # Where log archive will be stored
days_to_zip = 2                 # Compress files more than 2 days old
days_to_archive = 5             # Archive files more than 5 days old
admin_address = root            # e-mail address to use when reporting
                                # problems
email_wait = 10                 # do not send the same message more often
                                # than every 10 minutes
ignor_codes = DI                # do not log messages of levels D (debug)
                                # or I (informational). Use Z to store
                                # all messages.
```

Sample fbs.cfg

```
#------------------------------------------------------------------
# Define location for persistent job database and history database

%set jobdb
root = /home/farms/fbsng/jdb    # Job data storage directory

%set history
hist_dir = /home/farms/fbsng/history
                                # History database location
hist_file = history.log         # Current history file name
archive_file=archive.log        # Archived history file name
archive_period=1                # Move history to archive every 1 day

#------------------------------------------------------------------
# Define Launcher parameters. Use "%set launcher" for default
# parameters and "%set launcher <node-name>" for node-specific
# parameters

%set launcher
stat_port = 3333                # Run-time status information server
                                # UDP port number
log_ignore = D                  # do not send debug messages to logd

# template for command to be used to create Kerberos credentials
# for batch processes
k5_kinit_command = kinit -k -r 96h -t /var/adm/krb5/%u.keytab
  %u/prototype/farm@FNAL.GOV

# If necessary, define different parameter values for different nodes
%set launcher h1
stat_port = 5555
k5_kinit_command = kinit -k -r 1h -t /var/adm/%u.keytab
  %u/prototype/farm@FNAL.GOV

#-----------------------------------------------------------------
# Configuration of Kerberos-based FBSNG user authentication

%set kerberos
client_auth_required = yes             # authentication is required
principal = fbs/fnpcb.fnal.gov         # FBSNG service principal name
keytab = /home/farms/fbs.keytab        # location of service keytab file

%set user_profile                      # default user profile
principals = %u@FNAL.GOV               # require user's credentials
create_k5_credentials = optional       # create credentials if you can

%set user_profile farmprod             # group account
principals = alice@FNAL.GOV            # allow Alice and Bob to use this
    bob@FNAL.GOV bob/*@FNAL.GOV        # account
create_k5_credentials = required       # credentials must be created

%set user_profile guest                # guest from another realm
principals = + %u@CERN.CH              # trust CERN Kerberos
    guest/prototype/farm@FNAL.GOV      # allow submission from a farm job
```

Appendix C: Sample farm.cfg file

```
#---------------------------------------------------------------------------
#- File: farm.cfg
#- FBSNG farm configuration file.
#-
#- IMPORTANT NOTICE:
#- =================
#- This file should be edited directly only if absolutely necessary.
#- Use "fbs config" or "fbs monitor" to modify farm configuration.
#- If editing is necessary, preserve header lines starting with "#-"
#- and modification history lines starting with "##".
#-
#- farm.cfg set description:
#- ------------------------
#- %set resources
#- local = <local resource name> ...
#- global = <global resource name>:<integer capacity> ...
#-
#- %set resource_pools
#- <pool name> = <resource name> ...
#- ...
#-
#- %set node_class <class name>
#- resources = <local resource name>:<capacity> ... <attribute> ...
#- local_disks = <resource name>:<root directory> ...
#-
#- %set node_list
#- <node name> = <node class name>
#- ...
#-
#- %set proc_type <proc.type name>
#- proc_rsrc_defaults = <resource name>:<amount> ... <attribute> ...
#- sect_rsrc_defaults = <global resource name>:<amount> ...
#- resource_quota = <resource name>:<max. amount> ...
#- max_prio_inc = <integer max. allowed priority increment>
#- users = (-|*|<username> …)
#-
#- %set queue <queue name>
#- s_prio_gap = <section priority gap>
#- proc_type = <default process type>          (required field)
#- q_prio_max = <max. queue priority>
#- q_prio_gap = <queue priority gap>
#- s_prio_max = <max. section priority>
#- q_prio_min = <min. queue priority>
#- q_prio_inc = <queue priority increment>
#- q_prio_dec = <queue priority decrement>
#- status = [locked] [held]
#- users = (-|*|<username> …)
#- proc_types = (-|*|<process type> …)
#-
#----------------------------------------------------------------------------

## Fri Sep 15 11:38:33 2000: ivm: removed node fnpc111
##
## Fri Sep 15 11:38:33 2000: ivm: removed node fnpc112
##

%set node_list
fnpc104 = Worker
fnpc105 = Worker held farms: for system upgrade
fnpc106 = Worker
fnpc107 = Worker
fnpcb = IO
fnpc102 = Worker
fnpc103 = Worker
fnpc108 = Worker
fnpc101 = Worker

%set resources
```

Sample farm.cfg

```
local = IO cpu disk1 disk2
global = nfs_disk:30

%set proc_type IO
resource_quota = cpu:1000
proc_rsrc_defaults = IO cpu:50

%set proc_type Light
resource_quota = cpu:150
proc_rsrc_defaults = cpu:1

%set proc_type Worker
proc_rsrc_defaults = cpu:100

%set node_class IO
resources = disk2:20 IO cpu:200
local_disks = disk2:/tmp/fbs_scratch

%set node_class Worker
resources = cpu:100 disk1:10
local_disks = disk1:/tmp/fbs_scratch

%set resource_pools
disk = disk1 disk2

%set queue FastQ
q_prio_max = 100
s_prio_max = 100
q_prio_gap = 5
s_prio_gap = 1000
proc_type = Worker
q_prio_inc = 3
q_prio_min = 2
users = xyzprod xyzadmin

%set queue IOQ
s_prio_gap = 1000
proc_type = IO
q_prio_max = 1000
q_prio_gap = 1000
s_prio_max = 100
status = locked held
proc_types = IO


%set queue LongQ
s_prio_gap = 10
proc_type = Worker
q_prio_max = 100
q_prio_gap = 10
s_prio_max = 100
users = *
```